NYC Taxi Data Pipeline Document

Tran Trung Hieu
04/2025

Revisions

Version	Description of Version	Date Completed
1.1.0	First Complete	5/3/25

Contents

1. Introduction

1.1 Overview

This project focuses on building a big data pipeline to process NYC Taxi data. The pipeline includes both batch and streaming components to analyze taxi trip data for business insights, such as payment trends and customer behavior, while also generating synthetic streaming data for scalability.

1.2 Objectives

Gain hands-on experience with big data tools (Spark, Hadoop, Kafka, Hive, etc.) to deepen understanding of distributed systems.

2. Data & Business Requirements

2.1 Data Source

The data source is collected from the NYC Taxi and Limousine Commission (TLC) every month. However, the original data lacks diversity and customer details (e.g., phone numbers, names). Synthetic data is generated to address this, scaled to 1.7 GB per month (green and yellow taxi), with added fields for customer information.

The dataset includes:

- **VendorID**: Taxi provider ID
- **tpep_pickup_datetime**: The date and time when the meter was engaged.
- **tpep_dropoff_datetime**: The date and time when the meter was disengaged.
- **passenger_count**: Number of passengers.
- **trip_distance**: Distance in miles.
- **RatecodeID**: Fare rate code
- **store_and_fwd_flag**: Store-and-forward flag
- **PULocationID**: TLC Taxi Zone in which the taximeter was engaged.
- **DOLocationID**: TLC Taxi Zone in which the taximeter was disengaged.
- **payment_type**: Payment method
- **fare_amount**: The time-and-distance fare calculated by the meter
- **extra**:Miscellaneous extras and surcharges.
- **mta_tax**: Tax that is automatically triggered based on the metered rate in use.
- **tip_amount**: Tip amount – This field is automatically populated for credit card tips. Cash tips are not included.
- **tolls_amount**: Total amount of all tolls paid on the trip.
- **improvement_surcharge**: Improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
- **total_amount**: The total amount charged to passengers. Does not include cash tips
- **congestion_surcharge**: Total amount collected in trips for NYS congestion surcharge.
- **Airport_fee**: For pick up only at LaGuardia and John F. Kennedy Airports.
- **cbd_congestion_fee**: Per-trip charge for MTA's Congestion Relief Zone starting Jan. 5, 2025.
- **phonenumber**: Customer phone number.
- **full_name**: Customer name.

Streaming Data

Streaming data is simulated to mimic real-time taxi trips using a Python script that publishes to Kafka. It uses historical data distributions (e.g., pickup locations, trip distances) and adds customer details from a customers.csv file.

- **Frequency**: 2 trips per second

3Vs Overview

3V	Batch Pipeline	Streaming Pipeline
Volume	1.7 GB/month (green + yellow taxis)	
Velocity	Ingested in monthly batches	2 records/sec
Variety	20+ fields: timestamps, fares, surcharges, taxes, payment, customer phone & name, etc.	Subset of fields (6 fields): pickup time, location IDs, passenger count, trip distance, rate code, customer phone

More Info About Dataset:

- **Original Data Source:** https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page
- **Data Dictionary:** https://www.nyc.gov/assets/tlc/downloads/pdf/data_dictionary_trip_records_yellow.pdf

2.2 Business Requirements

The pipeline addresses two business needs:

Batch Pipeline: Payment Trend Analysis

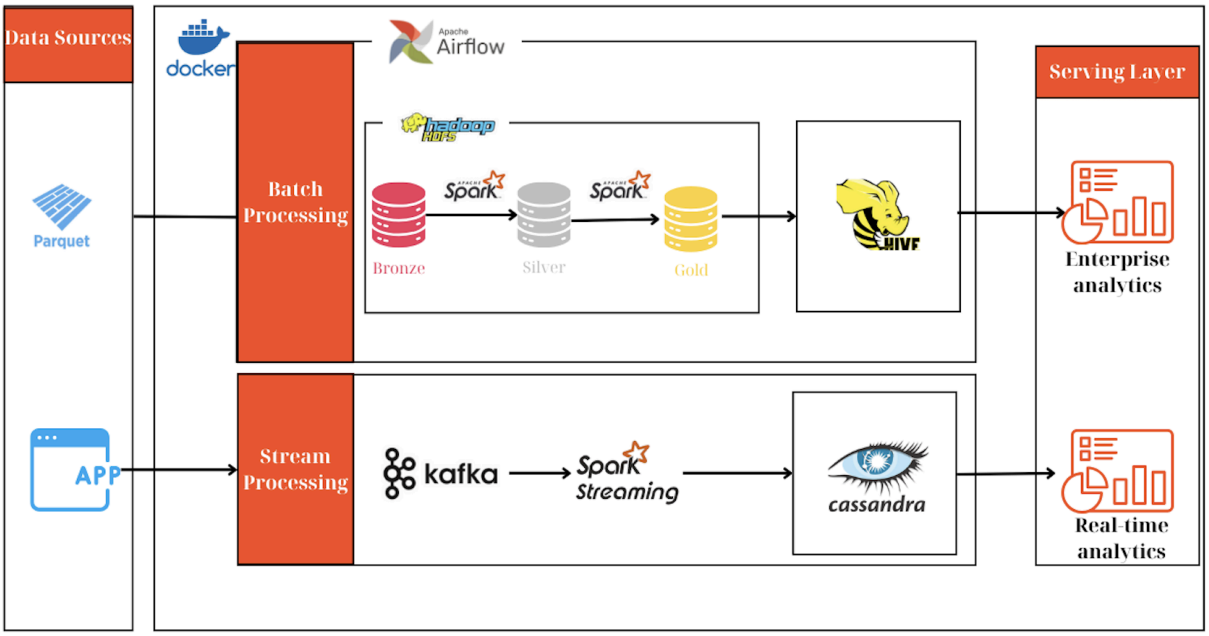
- **Objective**: Analyze payment trends to encourage credit card usage by offering fare discounts for credit card payments, focusing on customer behavior by day of week and fare range.
- **Focus Area:**
  - Credit- vs. cash-payment share on weekdays vs. weekends
  - Fare-amount bands and their payment mixes

Streaming Pipeline: Real-time Zone Aggregation

- **Objective**:
  - Ingest live taxi‑trip events, compute 15-min sliding-window pickup counts per zone in real time, and write results to Cassandra.
  - *Downstream consumers (out of scope for this project)* will then use this data to balance supply vs. demand.
- **Key Metrics:**
  - 15-minute sliding-window trip counts, refreshed every 5 minutes

3. Architecture and Data Modeling

3.1 Architecture



Component Details:

a) Hadoop HDFS

Overview: The primary storage system for the Batch Layer, hosting the Bronze, Silver, and Gold layers of the Medallion model to support **payment trend analysis** and ensure **long-term scalability.**

Medallion Layers:

- **Bronze**: Raw monthly Parquet files for each taxi service (yellow, green) stored unchanged.
- **Silver**: Cleaned, type-cast, and merged trip data with a new service_type column.
- **Gold**: Star-schema warehouse with multiple dimension tables and a year/month-partitioned fact_trips table.

Benefit:

- Scalability: The 1.7 GB/month dataset scales via additional disks/nodes for multi-year data or new sources (e.g., Uber/Lyft) without migration.
- Silver and Gold layers minimize repetitive processing.

b) Apache Spark

Why Spark in this project?

- One Engine, Two Modes – Same PySpark API drives batch jobs (Silver / Gold transforms) and the cash-payment stream, cutting dev overhead.
- Provides hands-on experience with PySpark for data processing, though the small dataset limits opportunities to fully practice optimization techniques (e.g., advanced partitioning, caching) and leverage Spark's distributed computing power.

c) Hive Layer

This layer stores references to external Parquet tables from the Gold layer in HDFS. The Fact table is partitioned by year/month; new partitions are added as data lands. Trino is used as the query engine due to a misconfigured Hive engine.

d) Apache Airflow

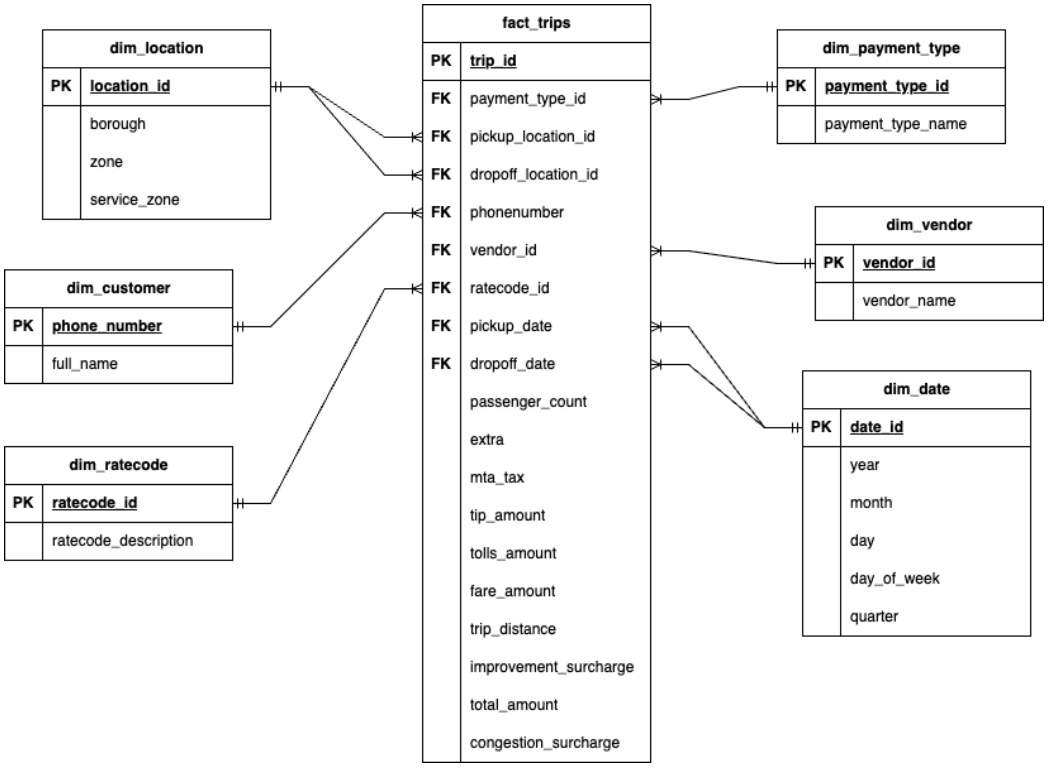
Airflow is used to:

- Orchestrate ETL workflows (data ingestion, transformation, querying) across Spark, Hive, etc.
- Schedule monthly runs at the start of each month to process the previous month's data
- Leverage flexible operators (SparkSubmitOperator, BashOperator, HiveOperator) for seamless task execution

e) Streaming Components

- **Kafka (Message Broker):** High-throughput, durable broker for live taxi events (2 msg/s) – decouples producers/consumers, persists events during Spark downtime, and supports replay.
- **Spark Structured Streaming:** Real-time sliding-window (15 min window, 5 min slide) counts by zone using the same Spark API as batch – ensures correct, sub-2 s micro-batch latency.
- **Cassandra:** Stores 15 min sliding-window pickup-count aggregates by zone with 48 h TTL – a horizontally scalable wide-row datastore optimized for high-throughput writes and low-latency (< 10 ms) reads, ideal for real-time dashboards.

3.2 Data Modeling:



Data Modeling Overview

The schema follows a star design with a central fact_trips table containing all trip metrics (fares, distances, passenger counts, etc.) and linked to date, payment type, location, vendor, and customer dimensions.

Supporting Two Separate Use Cases

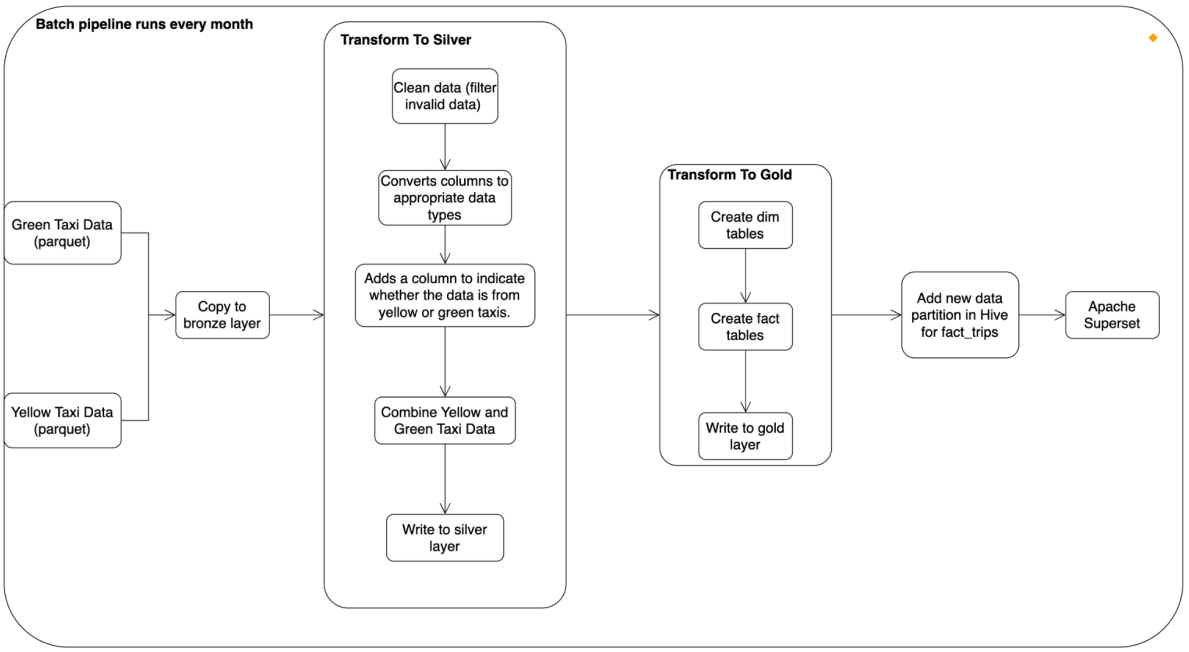
- **Fare-Band Cash Ratio:** Query fact_trips joined to **dim_payment_type** and **dim_date**, group by defined **fare_band** buckets to compute the share of trips paid in cash.
- **Cash Trips by Weekday:** Filter fact_trips for **payment_type** = "Cash", join **dim_date**, group by **day_of_week** and count trips per day.

Extensible Analytics

This star schema not only delivers the **cash-payment** and **weekday-analysis** insights but also **serves** as a **flexible foundation** for any **future analytics needs**.

4. Taskflow Details

4.1 Batch Pipeline

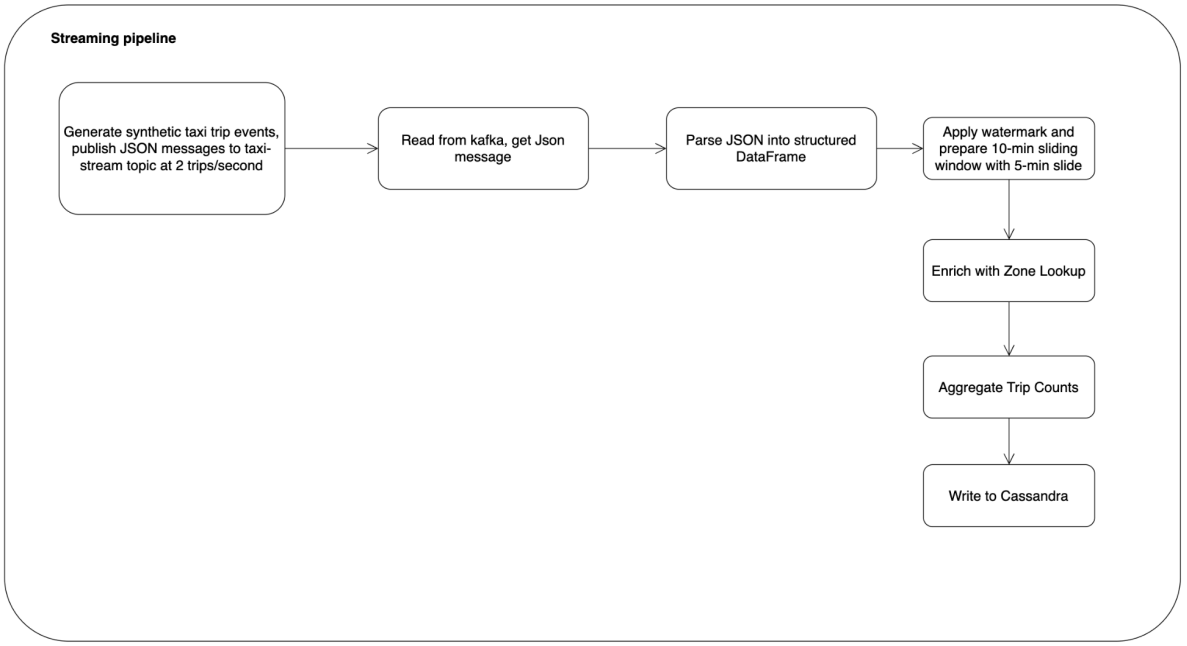


Business Problem: Cash payments are less efficient for taxi operators (e.g., slower transactions, higher risk of errors). By identifying patterns in payment behavior, the business can offer incentives (e.g., 5% discount for credit card payments) to shift customers toward digital payments.

How the Pipeline Supports It:

- The batch pipeline processes monthly taxi data (yellow and green taxis) through Bronze (raw), Silver (cleaned, unified), and Gold (star schema) layers.
- The Gold layer's fact_trips table, linked to dimension tables (e.g., dim_payment_type, dim_date), enables SQL queries to analyze:
    - **Credit vs. Cash Share**: Percentage of trips paid by credit card vs. cash on weekdays vs. weekends.
    - **Fare Bands**: Payment by cash distribution across fare ranges (e.g., $0-10, $10-20, $20+).
- Hive external tables, queried via Trino, provide a data warehouse for these analytics.

4.2 Streaming Pipeline



Business Problem: Taxi supply and demand vary by location and time. Real-time data on pickup activity helps operators dispatch drivers to high-demand zones, reducing wait times and increasing efficiency.

How the Pipeline Supports It:

- The streaming pipeline ingests live trip events from Kafka (taxi-stream topic, 2 trips/second).
- Spark Structured Streaming processes the data:
    - Parses JSON messages.
    - Enriches with zone information from taxi_zone_lookup.csv.
    - Aggregates trip counts by zone and 15-minute windows (5-minute slides).
- Results are written to Cassandra (zone_counts table) for low-latency access.

5. Reporting

I have created 2 reports mainly for:

- Show how the share of cash transactions varies across different fare brackets to pinpoint the price bands where credit-card incentives will have the biggest impact.
- Display the volume of cash trips for each weekday to identify which days see the highest cash usage and time promotional efforts accordingly.